IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

Improve Security in a Software Development Life Cycle

Ruchi Dabey[*1], Vishnu Kumar[2]
[*1,2,] Suresh Gyan Vihar University, India
rushi.dabey@gmail.com

Abstract

Software Security is an essential dimension of software quality and should be part of an overall effort to consistently measure and improve the quality of the software development and integration practices. Effective Software Security Management has been emphasized mainly to introduce methodologies which are Practical, Flexible and Understandable. Software Security describes the need and methodology of improving the current posture of Application Development by integrating Software Security. It attempts to provide an effective platform for organizations to understand how they can align software security in their SDLC.

Security brings value to software in terms of people's trust. The value provided by secure software is of vital importance because many critical functions are entirely dependent on the software. That is why software security is a serious topic which should be given proper attention during the entire SDLC, 'right from the beginning'. Security is an important property of any software. Many applications are outsourced too where the application development lacks strong integration of software security. The growing need to address software security measures across development life cycle. Application Security can be seamlessly integrated in the SDLC by introducing specific steps or process within the development phases.

Key words: - Security rules, integration security with SDLC, Threat modeling, code review, education & training.

Security Rules

The various issues encompassing software security is a point of discussion and debate among the researchers and security practitioners. One obvious way to spread software security knowledge is to train software development staff on critical software security issues. Beyond awareness, more advanced software security training should offer coverage of security engineering, design principles and guidelines, implementing risks, design flaws, analysis techniques, and security testing.

All software developer must obey these rules in order not to introduce vulnerabilities into the system and ensure the production of secured software system. By analyzing the implementation results, it is observed that if the software engineers have these rules at the back of their minds throughout the stages of the software production, it will ensure efficient production of secure software product to a greater extent. These rules are given as follows:

1. Rule of Awareness
2. Rule of Prevention
3. Rule of Confidentiality
4. Rule of Integrity
5. Rule of Availability
6. Rule of Access Control
7. Rule of Accuracy
8. Rule of Consistency
9. Rule of Authorization
10. Rule of Privacy

Integration Security With SDLC

The main objective of proposed security model comes up with the following:

1. Well oriented- software development life cycle with security (stages that include development, usage, maintenance).
2. Well formulated requirement specification (both functional and non-functional requirement)
3. Well defined software development model (engineering model)
4. Well integrated with security (security include in designing, development and testing models)

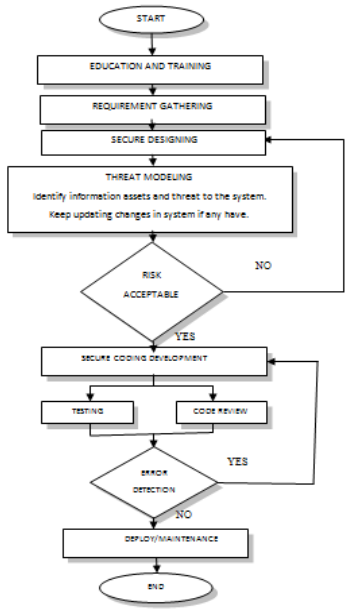
Most of the applications were development following process that is easily customizable to their management needs. Organization were unaware of the fact that security issue those were not addressed

properly during all these processed are going to fit them back with a much greater impact. The organization should have clear objectives, a team to complete or facilitate the completion of these objectives, and a formal, documented project plan describing the tasks required to attain Sustained Maturity. The next critical step is implementing the plan in a phased approach that will be easier to accommodate.

Secure application development, from design to maintenance, will be new ground and possibly a culture change. Security brings value to software in terms of people's trust. The value provided by secure software is of vital importance because many critical functions are entirely dependent on the software. That is why security is a serious topic which should be given proper attention during the entire SDLC, '*right from the beginning*'. Implementing security in software from the very stages of its development makes the system as vulnerable and faults free as possible. It also provides mechanism for quick recovery by the system from the damages caused by failure. It ensures that the system continues to operate under most adverse condition created due to the various attacks on the system. In doing so, the system provide a mechanism of resistance against the attacker who tries to exploit the weakness in the software. It also provides a tolerance level of such failures resulting from such exploits.

Work Flow of SDLC With Security Features

We can describe a diagram that show work flow of software development life cycle with advance security features.

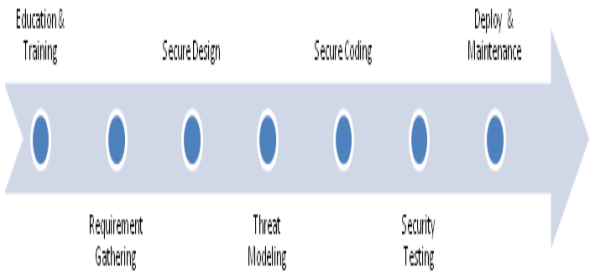


Proposed Security Model In SDLC

Let us discuss the following model to align application security in the SDLC. Several well-known Software Development Life Cycles (SDLCs) integrate security in different ways. Although many versions of this model exist, the SDLC generally starts with a requirements specification phase followed by design, implementation, testing, deployment and maintenance phases. Security can be integrated into any (and ideally all) of these phases. In most organizations security is included with the toll gate style mentioned previously, often at the end of each phase before moving to the next one. It is, however, critically important to ensure that security is prioritized during the requirements specification phase and carried out at every phase, particularly in organizations where developers and QA teams are responsible for policing themselves. If security drives the software development life cycle process, the responsible parties must work with developers to determine their needs and provide input during every step. This process enables the applications to be built securely while helping development teams to maintain a reasonable schedule. The seven principles should be considered design principles when developing or enhancing an existing software security assurance program.

The Framework addresses the following key component areas:

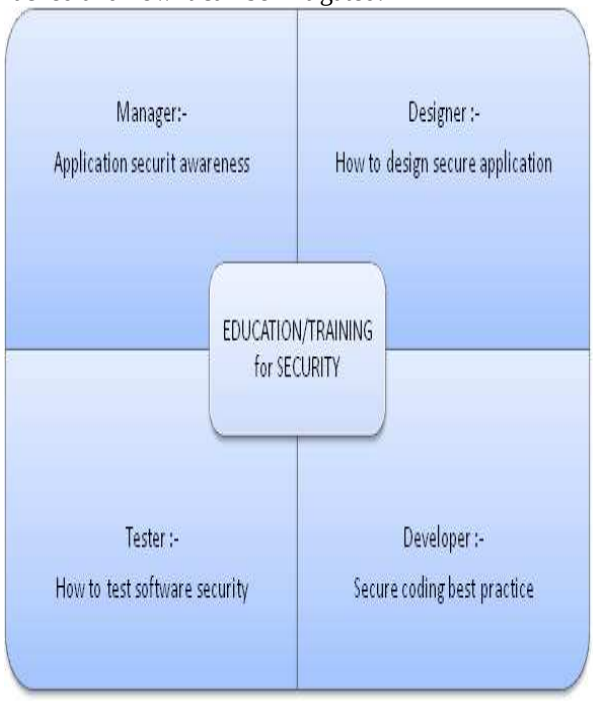
1. Education & Training
2. Requirement Gathering
3. Secure Design
4. Threat Modelling
5. Secure Coding Practices
6. Security Testing
7. Deploy & Maintenance



Education And Training

Awareness amongst the team members regarding application security issues is a must. The professionals must be able to foresee and address application security. You can think of having

different training courses to train your team to inculcate culture of application security throughout the lifecycle. You can also think of getting technology specific training courses to help developers understand and tackle issues easily. Educating the team on application security to help build resilience in them to withstand the adverse attacks and also think about them prior to designing applications. One of the better ways of training developers is demonstrating the vulnerabilities or exploits in their application or dummy application to help them realize how attackers work on. This perspective is very necessary to bring security in development phase. It helps them evaluate any module they work on with a mindset of how it can be hacked and how it can be mitigated.



The Secure Software Implementation identifies the essential components and principles of a mature software security program. A primary criterion for a successful program rests with the level of influence and change in behavior of application developers and systems integrators. There are many complex technical challenges in deploying vulnerability detection and management capabilities within a software security program for medium and large organizations, but all successful programs share a common commitment to education and training. An effective measurement of program maturity is to poll application developers to determine if they know the difference between software vulnerability and a software defect. If the majority of developers answer that vulnerabilities and defects are one in the same, then the education program can be considered successful. In fact, this is the single moment of truth for developers embracing secure development practices.

Requirement Gathering

Separate Security Requirements from Functional Requirements to facilitate explicit review and testing. Gather all the possible application security requirements from the customer. The customer may not be aware of the security risks, risks to the business objective and vulnerabilities that could creep in while building an application. In these cases, ask the customer for security requirements. Map specific security objectives in order to derive a secure design. The security objectives are based on understanding of what risks the end user might be exposed to and the risk to customer brand if security is compromised. For e.g. the objective that – only authorized users can access the application is the objective. This objective should be mapped to series of requirements. In the above example, a password protection mechanism could be implemented that enforces strong passwords (at least eight characters in length with a mandatory mix of numeric, upper case, lower case and special characters). The user account can be locked on five consecutive failed logon attempts.

Secure Design

There are two security activities that take place in this phase. The first security activity is to perform a more detailed risk assessment for each major system function and for the overall system. Based on this detailed risk assessment, security controls are selected to mitigate the risks identified. For example, because there is a high risk that backup media can be lost or stolen, the design team specifies that encryption is applied to the backup media to mitigate this risk.

The most effective practices in improving security architecture typically center on minimizing the publicly available application features. This principle is often called as reducing the surface area of attack. Document the assumptions and identify possible known attack scenarios against the system. Produce combined formal application specifications and security requirements specification. A much known activity adopted during this phase is called Threat Modeling. This is a technique used to understand and analyze threats against the system before it is built. It is primararily intended to identify architectural flaws rather than code level flaws. The basic steps in a Threat Modeling activity include Decomposing of Application, Defining Application Components & External Dependencies and Modeling the System to resolve threats. Review the design of the application

from security standpoint by enumerating all possible avenues of attack. This activity is genuinely helpful and does not have to be difficult. There are several threat, attack and vulnerability modeling tools and techniques. Microsoft in particular has emphasized Threat Modeling and had also provided a tool to enable modeling. The aim is to identify threats against each of the use case scenarios, system processes, data, transactions and functions. The common threats uncovered include possible loss in confidential data, unauthorized access, possible denial of service attacks, etc. By identifying all inappropriate actions that could be taken, we would capture all actions of malicious system use. This would help in mitigating the risks associated with the malicious system use. The risk response could be one which removes the risk, reduces the risk, transfers the risk or accept the risk.

Threat Modeling

It's common for security teams to receive reports of vulnerabilities with requests for immediate action to eliminate them. One big source of these requests is an organization's internal audit team. Another common source of fix-it-now-because-the-press/vendor-says-it's-critical messages is management, including many IS Directors. But should all vulnerabilities be considered emergencies? Are all vulnerabilities worthy of your security budget dollars?

One of the basic tenets of risk management is that not every vulnerability presents a threat to a network. Only a vulnerability that can be exploited is a threat to business operations and information assets. Threat modeling helps to identify those vulnerabilities that are actually critical in the unique environment that is your network. The threat modeling process should:

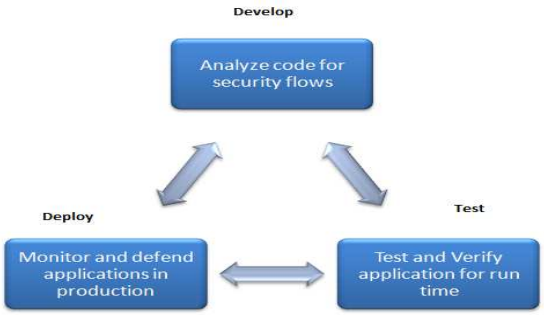
1. Identify potential threats and the conditions that must exist for an attack to be successful
2. Provide information about how existing safeguards affect required attack conditions
3. Provide information about which attack condition and vulnerability remediation activities add the most value
4. Help you understand which conditions or vulnerabilities, when eliminated or mitigated, affect multiple threats; this optimizes your security investment

Secure Coding

It is true that most of the security bugs sneak in the application during development phase. Security Issues during coding phase include the language choice, development environment, coding conventions, coding guidelines, baselines for security, documents for sensitive data handling, implementation of security features and integration with external applications or systems.

Establish Secure Coding Guidelines the process must mandate it for the developers to follow the guidelines for secure coding. There is a lot of information available for specific techniques for writing secure code on all different technologies. This information must be utilized to avoid coding errors. Ideally these practices are believed to be taught to the developers during the education phase. Hence, the important task here would be to ensure that these practices are followed across all the code modules and the defects are found at the early stage rather than the code being ready for deployment.

Many organizations take up premium tasks of using specialized tools for secure code review. Security Code Review tools available commercially are utilized to identify issues earlier in the cycle. However, automated code reviews are not very effective in analyzing all the security defects in the application. Hence critical parts of the applications must be provisioned for manual code review from peers or experts. Start rating the security defects to understand the impact and complexity of the issue. These metrics many a times helps you analyze the trade-off of performance because of intense security. Strike a balance between security and performance by verifying the business impact, complexity of attack and efforts required to mitigate the attack. The commonly accepted secure coding practices include data validation (input and output validation), segregation of trust, no hard coded secrets, proper error and exception handling, safe coding constructs to prevent known injection attacks and proper auditing and logging of application.



Secure Testing

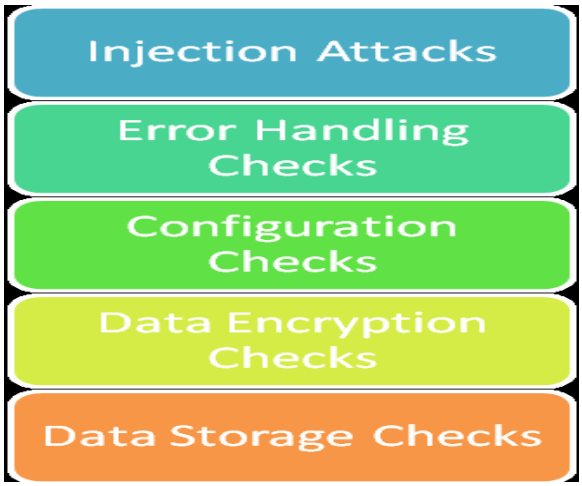
It is important to recognize that the cost of security increases as you move down the SDLC. Strategize to test early and testing often to prevent defects being leaked at a later stage and minimize the impact.

Organizations usually take up software security at this phase by adopting black box testing of applications to identify the potential risks. However,

the security bugs identified have a small window of closure and most of times end up in the production software by not being patched or having a temporary fix which may not be full proof. Black box testing has matured in very well in the industry.

The number of software code analysis or application scanners has increased exponentially in the market. Automated scanners provide vulnerability assessment or penetration testing. Security testing services are becoming a standard offering of many firms. Professional training for application security testers is widely available. One of the effective ways of handling security testing is to have a small team of security experts in house to take care of the testing or invest in an automated scanner to identify vulnerabilities. The option of hiring a third party to take penetration testing service may cost you premium.

Follow the security testing with a list of checks for the applications for comprehensive tests. The list must include the following checks at a minimum.



Deploy And Maintenance

It is observed many times that the development environment is lot more different than the production environment. The application in development environment is by default insecure because of many configurations in place like debugging enabled, trace enabled, accounts with known usernames and passwords, backup files present in the directories, etc. These are potential vulnerabilities found in the production boxes often categorized as Deployment Issues.

Few of the configuration checks include:

Conclusion

Security and development teams *can* work together—they just need to look for common areas in which they can make improvements. Security teams focus on confidentiality and integrity of data, which can sometimes require development teams to slow down and assess code differently. At the same time, business units require developers to produce and revise code more quickly than ever, resulting in developers focusing on what works best instead of what is most secure.

This difference in focus does not mean that either side is wrong. In fact, both teams are doing exactly what they're supposed to do. However, in order to facilitate teams accomplishing both sets of goals (timely release of both functional and secure software) and working together more fluidly, changes to tools and processes are necessary.

To begin, the organization needs to make a commitment to code security and evaluate the tools that can help accomplish that goal. A combination of static and dynamic code analysis tools usually works best, although using multiple tools often costs more and requires more time for training and implementation. These tools, then, should be integrated into development preferably in a largely automated manner to avoid slowing down development cycles as much as possible. Security teams should be involved in bug report reviews from both kinds of tools, and a continuous feedback loop should be created that allows all stakeholders to participate in development projects as appropriate.

Although this process will take time, the benefits will manifest in the form of a significantly more secure application landscape.

References

[1] CERT Software Engineering Institute Software Assurance - http://www.cert.org/work/software_assurance.html

[2] Microsoft® Security Development Lifecycle http://www.microsoft.com/security/sdl/default.aspx

[3] The Open Web Application Security Project (OWASP) – https://www.owasp.org/index.php/Main_Page

[4] The Build Security In Maturity Model (BISSM) - http://bsimm.com/

[5] *Building Security in Maturity Model* (version 2) May 2010. Gary McGraw, Ph.D., Brian Chess, Ph.D., & Sammy Migues

[6] *Secure and resilient software, requirements, test cases, and testing methods*. (2011). Merkow, M. S., & Raghavan, L., Auerbach Publishers

[7] Joseph Migga Kizza: A Guide to Computer Network Security, Springer, 2008, pp112-115.

[8] http://en.wikipedia.org/wiki/Timeline_of_computer_

[9] Security_hacker_history

[10] Jari Råman: Regulating Secure Software Development. Analysing the potential regulatory solutions for the lack of security in software, Lapland University Press, 2006, pp 2.

[11] Hao Wang, Andy Wang: Security Metrics for Software System, ACM Southeast Regional Conference, Proceedings of the 47th Annual Southeast Regional Conference, 2009, ACM-SE 47, pp 1-2.

[12] J. A. Wang, M. Xia, and F. Zhang, "Metrics for Information Security Vulnerabilities, Journal of Applied Global Research, Volume 1, No. 1, 2008, pp 48-58.

[13] http://www.executivebrief.com/project-management/software security-standards-project-security /P1/

[14] Julia H. Allen, Sean Barnum, Robert J. Ellison, Gary McGraw, Nancy R. Mead: Software Security Engineering: A Guide for Project Managers, Addison Wesley Professional, 2008, pp 6-8.

[15] http://www.isc2.org/uploadedFiles/(ISC)2_Public_Content/Certification_Programs/CSSLP/CSSLP WhitePaper.pdf

[16] http://cwe.mitre.org/documents/sources/Seven

[17] PerniciousKingdoms.pdf

[18] Oglund, G. and McGraw, G., Exploiting Software: How toBreak Code. Boston: Addison-Wesley, 2006 pp 1-4.

[19] http://searchwarp.com/swa268042.htm

[20] Gary McGraw, Software Security: Building Security In, Addison Wesley Software Security Series, 2006, pp 36.

[21] S. Sodiya, S. A. Onashoga, and O. B. Ajayi: Towards Building Secure Software Systems, Volume 3, 2006, pp 636 –645.

[22] Vladimir Golubev: Using Of Computer Systems Accountability Technologies in The Fight Against Cybercrimes, Computer Crime Research Center, downloadable from http://www.crimeresearch.

[23] org/library/Using.htm

[24] Neil Daswani, Christoph Kern, Anita Kesavan: Foundations of security What Every Programmer Needs to Know, APRESS, 2007, pp 44.

[25] http://www.albion.com/security/intro-4.html

[26] http://security.practitioner.com/introduction/infosec_2.htm

[27] http://www.yourwindow.to/informationsecurity/

[28] gl_confidentialityintegrityandavailabili.htm

[29] http://en.wikipedia.org/wiki/Access_control

[30] Howard, M., Lipner, S., *"The Security Development*

[31] *Lifecycle - SDL: A Process for Developing Demonstrably More Secure Software"*, Microsoft Press, 2006.

[32] Gregoire, J., Buyens, K., Win, B. D., Scandarioto, R., Joosen, W., *"On the Secure Software DevelopmentProecss: CLASP and SDL Compared"*, In Proceedings of the Third International Workshop on Software Engineering for Secure Systems, 2007.

[33] Beznosov, K., Kruchten, P., *"Towards Agile SecurityAssurance"* In Proceedings of the 2004 Workshop on New Security Paradigms, 2005.

[34] Beznosov, K., *"Extreme Security Engineering: On*

[35] *Employing XP Practices to Achieve 'Good Enough*

[36] *Security' without Defining It."*, First ACM Workshop on Business Driven Security Engineering, 2003.

[37] Flechais, I., Sasse, M. A., Hailes, S. M. V., *"A process for developing secure and usable systems"*, In Proceedings of the 2003

Workshop on New Security Paradigms, 2003.
[38] Grance, T., Hash, J., Stevens, M., *"Security*
[39] *Considerations in the Information System Development Life Cycle"*, NIST, Computer Security Division, NIST Special Publication 800-64, REV. 1, 2004.
[40] Swanson, M., etc, *"Security Metrics Guide for*
[41] *Information Technology Systems"*, NIST, Computer
[42] Security Division, NIST Special Publication 800-55,2003.
[43] Agile Alliance, *"Manifesto for Agile Software*
[44] *Development"*, 2005, http://www.agilealliance.org.